CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

Branch-and-Price for Service Network Design with Asset Management Constraints

Jardar Andersen
Marielle Christiansen
Teodor Gabriel Crainic
Roar Grønhaug

December 2009

CIRRELT-2009-58

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone 514 343-7575
Télécopie 514 343-7121

Bureaux de Québec :
Université Laval
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone 418 656-2073
Télécopie 418 656-2624

www.cirrelt.ca

UNIVERSITÉ LAVAL · UQÀM · HEC MONTRÉAL · ÉCOLE POLYTECHNIQUE MONTRÉAL · Université de Montréal

Branch-and-Price for Service Network Design with Asset Management Constraints

Jardar Andersen[1,*], Marielle Christiansen[1], Teodor Gabriel Crainic[2], Roar Grønhaug[1]

[1] Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology, 7491 Trondheim, Norway

[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), and Department of Management and Technology, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal, Canada H3C 3P8

Abstract. We address the service network design problem with asset management considerations for consolidation-based freight carriers. Given a set of demands to be transported from origins to destinations and a set of transshipment facilities, the objective is to select services and their schedules, build routes for the assets (vehicles) operating these scheduled services, and move the demands (commodities) through the resulting service network as efficiently as possible. We propose a first branch-and-price framework for the mixed integer formulation of the problem with integer cycle-design and continuous flow-path variables. The proposed method includes particular column generation subproblems for dynamically constructing these cycles and paths, as well as an acceleration technique to identify integer solutions rapidly. The computational study shows that the proposed method finds better solutions for large network instances than reported previously.

Keywords. Service network design, cyclic schedules, asset management, branch-and-price, column generation.

Acknowledgements. This work has received financial support from The Norwegian Research Council through the Polcorridor Logchain project. Partial funding has also been supplied by the Natural Sciences and Engineering Research Council of Canada (NSERC) through its Discovery and Industrial Research Chair programs.

Results and views expressed in this publication are the sole responsibility of the authors and do not necessarily reflect those of CIRRELT.
Les résultats et opinions contenus dans cette publication ne reflètent pas nécessairement la position du CIRRELT et n'engagent pas sa responsabilité.

---

* Corresponding author: jan@toi.no

Dépôt légal – Bibliothèque et Archives nationales du Québec,
Bibliothèque et Archives Canada, 2009

© Copyright Andersen, Christiansen, Crainic, Grønhaug and CIRRELT, 2009

Introduction

Improvements in solution techniques and computational capacity continuously increase the range of tractable optimization problems. Simultaneously, real-world planning problems challenge the Operations Research community to study increasingly larger and more complex optimization problems. New requirements from practical applications result in new constraints that have to be modeled and trigger the development of solution methods to address these new models.

We focus on *scheduled* service network design problems for consolidation-type carriers, where services represent transportation operations defined in terms of origin and destination terminals, route, speed and capacity. The major decisions are selecting the services and their departure times, as well as routing of origin-to-destination commodity loads through the service network, such that a given demand for transportation is satisfied at minimum total system cost. Such problems are generally modeled as mixed integer formulations on time-space networks, where binary design variables select the scheduled services, while continuous variables represent commodity flows through the resulting service network; see, e.g., the reviews of Christiansen *et al.* (2007) for maritime transportation, Cordeau *et al.* (1998) for rail transportation, Crainic (2003) for long-haul transportation, Crainic and Kim (2007) for intermodal transportation, and Crainic (2000) for service network design in freight transportation.

The class of Service Network Design planning problems with *Asset Management* (SNDAM) has attracted increased interest recently. *Asset* is a generic term for something a person or an organization owns. In carrier terms, it generally refers to what is also sometimes called resources: power units (tractors, locomotives, etc.), carrying units (railcars, trailers, containers, ships, etc.), loading/unloading units (cranes in terminals), crews, and so on. The goal of these models is to integrate operational concerns into tactical planning. The literature and the problem considered in this paper address the management of one type of asset only. In the rest of the paper we therefore refer to assets as "vehicles".

Most service network design models with vehicle management presented in the literature focus on enforcing the requirement of an equal number of services entering and leaving each node (terminal) in the network, see for instance Pedersen and Crainic (2007), Pedersen *et al.* (2009), Smilowitz *et al.* (2003), Lai and Lo (2004), Barnhart and Schneur (1996), and Kim *et al.* (1999). This is achieved through *design-balance constraints* (Pedersen *et al.*, 2009) in problem settings where it is assumed that each service requires a single vehicle to operate. Andersen *et al.* (2009a) assumed the same hypotheses, but introduced additional aspects of vehicle management, including the management and coordination of multiple fleets and fixed-length, cyclic schedules, i.e., schedules operated repetitively over a certain period of time (e.g., a season). They denoted this problem the Service Network Design with Asset Management constraints problem (SNDAM), which is the problem setting used in this paper as well.

The design of scheduled services can be formulated using either binary service design arc variables or cycle-design variables (Pedersen *et al.*, 2009), see also (Sigurd *et al.*, 2005) and (Agarwal and Ergun, 2008) for ship scheduling modeling. Similarly, for the flow of commodities one can either make use of arc flow variables or path flow variables. Andersen *et al.* (2009b) studied the four alternative formulations of SNDAM obtained from all combinations of the arc and cycle-design variables with the arc and path-flow variables for commodities. Their computational study indicated that cycle-design and path-flow variables contribute to efficient model solving. The focus of Andersen *et al.* (2009b) was on model development, however, and their computational study was based on a priori enumeration of cycles and paths. This approach is obviously not appropriate for problem instances of realistic dimensions. There is thus a need for more advanced solution strategies for this class of problems. The goal of this paper is to address this need.

We present a branch-and-price (B&P) solution method for the SNDAM problem, which relies on a decomposition of the problem into 1) a master problem, handling a variant of the multicommodity

network design problem with vehicle management constraints, and 2) two types of subproblems for generating design-cycles and path-flows, respectively. The outcome is a tailormade column generation-based method with branching in both the master problem and the subproblems. An acceleration technique is introduced to identify integer solutions rapidly. Different implementations of the B&P algorithm are presented and compared, together with simpler approaches such as a priori generation of columns and use of column generation in the branch-and-bound tree root node only combined with a commercial MIP solver. A comparative analysis of variants of the B&P method is also presented. The major contribution of the paper is the first B&P framework for SNDAM, a method that is able to solve larger instances than presented earlier.

The outline of this paper is as follows. In Section 1, we present the SNDAM formulation. In Section 2, we propose a B&P algorithm for solving the SNDAM. The computational study is presented in Section 3, while concluding remarks follow in Section 4.

1 Service Network Design with Asset Management

We describe in this section the Service Network Design problem with Asset Management constraints, SNDAM, presented in (Andersen *et al.*, 2009b). The problem description follows in Section 1.1, while the SNDAM model with cycle-design and path-flow variables is given in Section 1.2.

1.1 Problem Description

SNDAM is a time-dependent service network design problem (Crainic, 2000), where major decisions are the selection of services and their departure times, and the routing of demand through the selected service network. We assume that the *schedule length* is given, and that the services and their schedule are to be operated in a repetitive, *cyclic* manner representing the fixed schedules of real-world transportation services.

Demand is defined in terms of commodities (products) requiring transport through the network. Each commodity has an associated volume to be transported from the commodity's origin node to its destination node. In the problem setting we consider, commodities have specific times when they become available, but they may arrive at their destinations at any time, as long as it is within the length of the schedule.

Operation of services requires *vehicles*. In the current problem setting, a single vehicle is required for each service. The route a vehicle performs is a cycle composed of a series of services, waiting at terminals, and, eventually, repositioning movements. Vehicles are in limited supply and are subject to a rich set of management constraints (Andersen *et al.*, 2009b):

An equal number of services and vehicles enters and leaves a terminal at any moment.

The fleet size restricts the number of vehicles operating simultaneously.

There are minimum and maximum limits on the number of *occurrences* (*departures*) of each service in the schedule, e.g., a service should be operated at least 3 times and at most 7 times each week.

The duration or *length* of vehicle routes is limited to the length of the schedule.

Most service network design models in the literature assume fixed *service-selection costs* and unit *flow costs* associated with moving commodities using the selected services (Crainic, 2000). The fixed service costs are usually derived from the cost of operating the associated vehicles. In the SNDAM problem we include vehicle associated costs only, since vehicle activities are explicitly considered. It is noteworthy, however, that including service-selection fixed costs does not introduce any structural changes to the SNDAM model presented in the next subsection.

1.2 The Cycle-Path SNDAM Model

We define a static, hereafter referred to as physical, network with nodes $i \in N$ representing terminals connected by services $s \in S$. To simplify the presentation, but without loss of generality, we do not consider services consisting of multiple arcs.

The schedule length is divided into a set of time periods $T = \{1,...,T\}$. We introduce a time-space network with one node i_t for each terminal $i \in N$ at each time period $t \in T$. Each service $s \in S$ may be operated at each of the T periods, and, thus, it is represented T times in the time-space network through a *service arc* for each potential departure time. The duration (length) of a service is given as a number of periods, the same for all its occurrences, while its total capacity is denoted U_s. *Holding arcs* link consecutive time realizations of the same terminal where vehicles and commodities may wait. Holding arcs have a length of one period and are assumed to have infinite capacity for both vehicles and flows. The time-space network is cyclic, that is, period T precedes period 1.

Figure 1 illustrates the cyclic time-space network for a five-terminal system over $T = 7$ periods. For clarity sake, only one occurrence for each of eight services and two holding arcs are shown. The grey service of length one from node 3 in period 7 to node 2 in period 1 and the black holding arc from period 7 to period 1 at terminal 4 illustrate the cyclic nature of the network.

Transportation services are provided by a homogeneous fleet of *vehicles* $v \in V$, of cardinality $V = |V|$. Any selected service departure uses one of these vehicles, LF_s and UF_s standing for the lower and upper bounds, respectively, on the number of departures one may select for service $s \in S$. We define a vehicle design-cycle $k \in K_V$ as a sequence of service and, possibly, holding arcs satisfying design-balance requirements and covering each time period exactly once (Requirements A and D). The cost of cycle k is denoted F_k, while M_{stk} takes value 1 if service $s \in S$ departing at time $t \in T$ is included in cycle k and 0 otherwise. No explicit vehicle repositioning arcs are included in the space-time network. To simplify the presentation and without loss of generality, service arcs without flow in the final solution represent the repositioning of vehicles (when needed). Figure 1 illustrates two vehicle routes supporting in total eight services and satisfying the requirements A-D described above. The figure also illustrates the fact that it is possible for a node to be visited by more than one vehicle at the same time.

With respect to demand, we define for each commodity $c \in C$ the volume W_c to be transported from its origin node to its destination node, which yields $B_{sc} = \min\{W_c, U_s\}$ as the upper limit on volume of commodity c that may use service s. Let P_c represent the set of paths that may be used to transport commodity c, where $A_{stcp} = 1$ if path $p \in P_c$ for commodity c uses service $s \in S$ departing at period $t \in T$, 0 otherwise. The unit flow cost on path p is denoted K_{cp} (we include both indices, c and p, to increase readability).

Two sets of decision variables are defined:

- Vehicle cycle-design variables: g_k takes value 1 if cycle $k \in K_V$ is selected and 0 otherwise;
- Commodity path-flow variables h_{cp} giving the volume of commodity c on its path p.

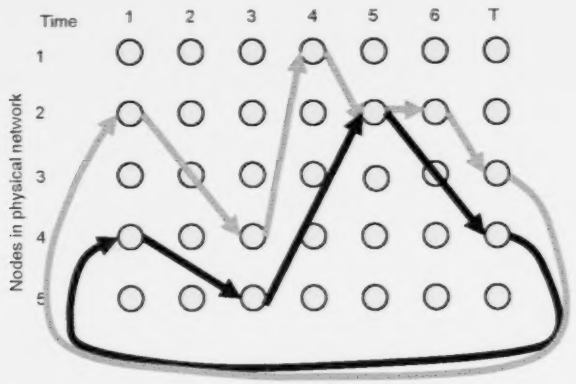

Figure 1. Time-space network with two vehicles operating.

The SNDAM model with cycle-design and path-flow variables is then formulated as follows:

$$Min\ z = \sum_{k \in \mathrm{K}} F_k g_k + \sum_{c \in \mathrm{C}} \sum_{p \in \mathcal{P}_c} K_{cp} h_{cp} \tag{1}$$

$$\sum_{k \in \mathrm{K}} g_k \leq V, \tag{2}$$

$$\sum_{k \in \mathrm{K}} M_{stk} g_k \leq 1, \qquad \forall s \in \mathrm{S}, t \in \mathrm{T}, \tag{3}$$

$$LF_s \leq \sum_{k \in \mathrm{K}} \sum_{t \in \mathrm{T}} M_{stk} g_k \leq UF_s, \qquad \forall s \in S\backslash, \tag{4}$$

$$\sum_{p \in \mathcal{P}_c} h_{cp} = W_c, \qquad \forall c \in \mathrm{C}, \tag{5}$$

$$\sum_{c \in \mathrm{C}} \sum_{p \in \mathcal{P}_c} A_{stcp} h_{cp} - \sum_{k \in \mathrm{K}} U_s M_{stk} g_k \leq 0, \qquad \forall s \in \mathrm{S}, t \in \mathrm{T}, \tag{6}$$

$$\sum_{p \in \mathcal{P}_c} A_{stcp} h_{cp} - \sum_{k \in \mathrm{K}} B_{sc} M_{stk} g_k \leq 0, \qquad \forall s \in \mathrm{S}, t \in \mathrm{T}, c \in \mathrm{C}, \tag{7}$$

$$h_{cp} \geq 0, \qquad \forall c \in \mathrm{C} y p \in \mathrm{P}_c, \tag{8}$$

$$g_k \in \{0,1\}, \qquad \forall k \in \mathrm{K}\backslash, \tag{9}$$

The objective function (1) minimizes the sum of the fixed costs for selected cycles and the flow costs on paths. Constraint (2) restricts the cycle selection to the number of available vehicles (Requirement B). Through constraints (3), we enforce that each service $s \in \mathrm{S}$ departing at time $t \in \mathrm{T}$ can be supported by at most one cycle (vehicle). These constraints (3) may take different forms for particular applications; the current formulation corresponds to the rail freight transportation planning problem (Andersen et al., 2009b) used to generate the test problem instances for this paper. Lower and upper bounds on the number of service occurrences are imposed in (4) (Requirement C), while demand satisfaction is ensured in (5). Weak and strong forcing constraints are given in (6) and (7), respectively, enforcing service capacity restrictions and forcing flows to zero on unselected service departures. Finally, variable-type constraints appear in (8) – (9).

The SNDAM model is particularly relevant and important for the transportation industry with consolidation-based freight carriers as for instance freight train services and liner shipping. For such problems there are relatively many commodities and time periods compared to the number of physical nodes, and therefore the SNDAM deviates from a large class of vehicle routing problems. Notice that, as for other network design formulations, the strong forcing constraints (7) are redundant and their introduction increases the size of the problem considerably. Yet, it is known that the strong forcing constrains may significantly strengthen bounds obtained from linear relaxations, and that they may be gradually added as valid inequalities during the resolution process. We follow this approach for SNDAM and discuss implementation and impacts in Section 3.

Magnanti and Wong (1984) showed that the uncapacitated fixed charge network design problem is NP -hard. Capacitated versions are even harder (Balakrishnan *et al.*, 1997) and also belong to the class of NP -hard problems. Incorporating additional constraints such as (2) – (4) adds further complexity to the problem. Even finding feasible solutions to SNDAM is far from trivial. In the case of the network design formulation, a (not necessarily good) feasible solution may be easily found by setting all arc design variables to one (opening all arcs) and solving the corresponding network flow problem. This is not the case for the SNDAM problem, where the cycle-design variables and constraints (2) – (4) prevent the trivial "all arcs open" solution to be feasible, except for special instances.

2 A Branch-and-Price Algorithm for SNDAM

Andersen *et al.* (2009b) generated a priori all design-cycle and flow-path variables for the SNDAM model (1) – (9). However, this approach suffers from poor scaling capabilities, and other solution methods are required to solve large problem instances. Pedersen *et al.* (2009) proposed a metaheuristic for a simpler problem setting (only design-balance constraints were included), but no exact method has been proposed yet. Column generation embedded within a branch-and-price framework appears as a promising exact solution method due to the path and cycle structure of the problem. We describe such a method in this section.

The *column generation approach* solves the linear relaxation of the model (1) – (9) constituting the master problem. The basic principle is to work on a *Restricted Master Problem* (RMP) that includes all the rows, but only a subset of the variables (or columns). Starting with a limited set of columns (artificial or real cycle and path variables), the method then iteratively adds new cycle and path columns to the RMP, providing they improve its objective function. The method continues to dynamically generate columns until no additional promising columns may be found. One can then conclude that the current RMP solution is an optimal solution to the linear relaxation of the original formulation (1) – (9) and a lower bound of its mixed-integer version. We use the simplex method to solve the RMP, which yields primal and dual solutions. This dual information is used to identify "new" columns with negative reduced costs in problem specific subproblems, and these columns are then added to the RMP. One subproblem generates cycles with negative reduced cost for the homogeneous fleet of vehicles, while another type of subproblems generates negative reduced cost path-flows for each commodity. The subproblems correspond to two types of shortest path problems that can easily be solved by dynamic programming.

Branching occurs when no new columns improve the RMP and the LP-relaxed solution does not satisfy the integrality conditions of the original formulation. Column generation is applied in every node in the search tree resulting in a *branch-and-price* (B&P) method. We impose branching decisions by modifying either the networks in the subproblems or the constraints (or bounds) in the master problem. Several column generation topics are covered in Desaulniers *et al.* (2005), and general introductions to B&P and column generation can be found in Barnhart *et al.* (1998) and Lübbecke and Desrosiers (2005).

The rest of this section is organized as follows: Section 2.1 discusses the RMP for the SNDAM formulation, while Sections 2.2 and 2.3 are dedicated to the subproblems for the design- cycle and path-flow generation, respectively. Branching strategies are discussed in Section 2.4, while an acceleration technique for upper bound computations is presented in Section 2.5.

2.1 Restricted Master Problem

The restricted master problem (RMP) is the linear relaxation of the SNDAM formulation (1) – (9) with some adjustments. We work on subsets of cycles and paths; $\tilde{K} \subseteq K$ and $\tilde{\mathcal{P}}_c \subseteq \mathcal{P}_c$, respectively. To obtain the linear relaxation, we replace the integrality requirements (9) with (9b). We also reformulate (4) to (4b) by introducing explicit slack variables, $q_s, \forall s \in S$, $0 \le q_s \le (UF_s - LF_s)$, which halves the number of constraints (4). Similarly, constraints (2) are reformulated to (2b) by introducing an explicit slack variable r for the difference between the fleet size and the actual number of vehicles in use, $0 \le r \le V$. We describe in Section 2.4 how these slack variables are used for branching. The objective function (1) and the constraints (3) and (5) - (8) remain the same, except for the reduced number of variables in the RMP, and domain-definition constraints are added for the slack variables:

$$\sum_{k \in \tilde{K}} g_k + r = V, \tag{2b}$$

$$\sum_{k \in \tilde{K}} \sum_{t \in T} M_{stk} g_k + q_s = UF_s, \qquad \forall s \in S, \tag{4b}$$

$$0 \le g_k \le 1, \qquad \forall k \in \tilde{K}, \tag{9b}$$

$$0 \le q_s \le (UF_s - LF_s), \qquad \forall s \in S, \tag{10}$$

$$0 \le r \le V. \tag{11}$$

We associate dual variables α, β_{st}, θ_s, σ_c, η_{st}, and ρ_{stc} to constraints (2b), (3), (4b), (5), (6), and (7), respectively. The dual information from the solution of the RMP is transferred to the subproblems for the generation of design-cycles and flow-paths. The RMP can be solved by use of a commercial LP-solver. The solution times may however be substantial for large instances. We hence solve the cycle and all the path generation subproblems for a given set of dual multipliers before returning to the RMP.

The restricted master problem (1), (2b), (3), (4b), (5)-(8), (9b), (10), and (11) yields the *strong linear relaxation* of the SNDAM formulation (1)-(9), the *weak linear relaxation* being obtained when the strong forcing constraints (7) are excluded. For large networks with many arcs and commodities, the number of strong forcing constraints grows extremely large. Consequently, we initially exclude the strong forcing constraints (7) from the formulation, and then, we generate dynamically those strong forcing constraints that are violated in the optimal solution of the RMP. To do this, we first solve the weak linear relaxation to optimality with column generation. Then, for each arc in the network having flow in the linear solution, we verify if any strong forcing constraints are violated for the products having paths traversing that arc. All such violated constraints are then added to the formulation. The new problem is then solved to optimality with column generation, followed by a new search for violated strong forcing constraints. This process is repeated until no strong forcing constraints are violated, and we have obtained the solution to the strong linear relaxation. When the strong forcing constraints are generated dynamically during the B&P, the resulting approach is denoted branch-and-price-and-cut.

The drawback of this approach is that for large instances, testing whether the strong forcing constraints are violated and reoptimizing after new constraints are added may become computationally expensive. An important issue is thus to which degree strong forcing constraints should be generated: at all branch-and-bound nodes, at the root node only, each time the lower bound is updated, or none of these. We address this issue in Section 3.

2.2 Design-Cycle Subproblem

A feasible design-cycle is a closed path that consists of a set of arcs satisfying the design-balance requirements, and not covering a time period more than once. The objective of the design-cycle subproblem is to find feasible cycles with minimum reduced costs to improve the objective function value of the current RMP. Let $\overline{CC_k}$, given in (12), represent the reduced cost of cycle variable g_k.

$$\overline{CC_k} = F_k - \alpha - \sum_{s \in S} \sum_{t \in T} \left(\beta_{st} - u_s \eta_{st} - \sum_{c \in C} B_{sc} \rho_{stc} + \theta_s \right) M_{stk} \tag{12}$$

The underlying network is the time-space network with additional artificial origin (s) and destination (d) nodes. We construct arcs from the artificial origin node to all physical nodes at time period 1. We also extend the network with nodes, denoted *end nodes*, for time period 1 beyond the planning horizon (the first period in the next repetition of the schedule), and link them to the artificial destination node, to provide the possibility to generate cycles through direct application of a shortest path routine. Additional *start nodes* and arcs need to be added to capture services with duration greater than one, since these may be part of cycles that are not visiting a node at time 1. For services with a duration of two time periods, one additional start node must be added at time T representing the origin of that service. For services with a duration of three time periods, a similar start node must be added at time $T-1$ in addition to the one at time T, etc. Then, one arc is added from the artificial start origin to each start node, and a service arc that "passes by" time period 1 connects the start node with the corresponding destination node of the service.

We illustrate these ideas in Figures 2 and 3. Figure 2 illustrates a tiny example of a network consisting of three physical nodes and three service arcs with the indicated time durations. The corresponding time-space network is given in Figure 3 with arcs from the artificial origin to all nodes in period 1, as well as to the start nodes of services starting in period 4 with duration superior to 1 (e.g., arc from node 3 at time 4 to node 2 at time 2, indicated by dotted lines). The figure also illustrates the end nodes, indicated with stripes, and the corresponding arcs to the artificial destination.

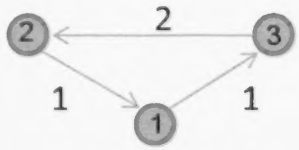

Figure 2. Example network with three physical nodes and arcs with associated travel durations.

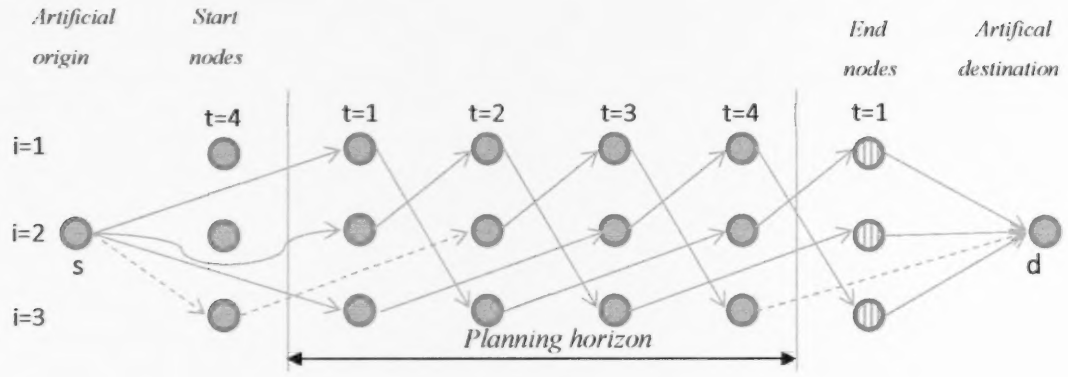

Figure 3. Time-space network for cycle generation.

The network is acyclic and the problem can be solved as a shortest path problem by dynamic programming (DP) in polynomial time. We use a label-correcting method with two labels. One label corresponds to the reduced costs for the partial path from the artificial origin node up to the present node. In addition, we need to ensure that the first physical node visited in a path is the same as the last physical node in the path in order to design a cycle. Hence, we introduce a label for this first physical node. Figure 4 illustrates two feasible cycles, where the dotted cycle uses the arc passing by time period 1.

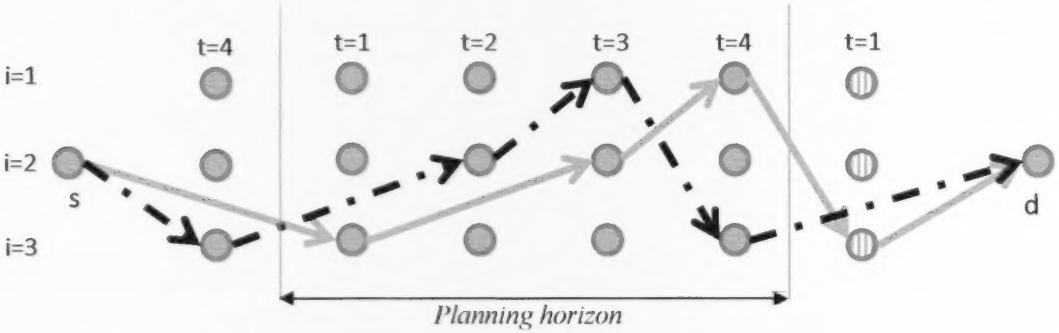

Figure 4.Two feasible cycles in the time-space network.

There exists just one subproblem for cycle-generation, but several cycles may exist in the final solution of the problem (1)-(9). The DP algorithm allows the generation of several cycles by starting from different first nodes.

2.3 Flow-Path Subproblems

A flow-path subproblem is solved for each commodity, the objective being to find a path with minimum reduced cost. The reduced cost $\overline{CP}_{cp}$ of variable h_{cp} is defined as follows:

$$\overline{CP}_{cp} = K_{cp} - \sigma_c - \sum_{s \in S} \sum_{t \in T} (\eta_{st} + \rho_{stc}) A_{stcp} \tag{13}$$

Paths start at the origin node of the commodity and must end at one of the time realizations of its destination node. Because the subproblem networks are acyclic, we can solve the shortest path problem by an efficient label-correcting DP algorithm in polynomial time. From model (1) – (9), we know that the dual variable σ_c is a free variable taking either positive or negative values. Note that dual variables η_{st} and ρ_{stc} cannot become positive. Hence, the subproblem can contribute a path with a negative reduced cost only if dual variable σ_c is strictly positive. Therefore, if at any stage the calculated reduced costs during the DP exceeds σ_c we can stop evaluating that path because it cannot result in a path with negative reduced costs. Similarly, because the reduced cost of a path increases with each arc, we can terminate it if the reduced cost at any stage is higher than the cost of a feasible origin-destination path.

2.4 Branching strategies for the integer problem

To obtain integer solutions for the SNDAM problem, we embed the column generation procedure into a branch-and-bound framework, where the solution obtained from the linear relaxation of the problem represents the *root node* of the search tree. At each node of the search tree, all existing feasible columns are kept and new columns are generated to obtain an optimal solution to the RMP. In this subsection, we present branching strategies for the exploration of the search tree. As is known from the B&P literature (e.g., Lübbecke and Desrosiers, 2005), it is not desirable to branch directly on the column variables. We consider instead three alternative branching strategies where we either alter the subproblem networks in order to generate feasible columns with respect to a branching decision, branch on the slack variables in the RMP, or modify the constraints in the master problem.

The first approach, BB-1, uses branching on service occurrences in the underlying network structure, i.e. arcs in the time-space network. In evaluating the solutions of the RMP, we choose the arc with fractional value closest to a threshold value, and fix this arc to 1 in one branch and to 0 in the other branch. In the 1-branch, we change the $\leq$ sign in constraints (3) to an = sign ensuring that the sum of the cycle variables using this arc is 1. When the branch is fixed to zero, all generated cycles using this arc are fixed to 0 as well. In addition, we remove this service arc from the underlying network in the subproblems. This branching strategy alone will lead to an integer solution, but a drawback is that it gives an unbalanced search tree. The decision to fix an arc to one gives a strong decision, while fixing it to zero results in a very weak decision with little impact on the subproblems. We always chose the 1-branch first.

The second branching strategy, BB-2, uses the slack variables q_s from (4b). For integer solutions to (1) – (9), the integrality of UF_s and of M_{stk} also ensures the integrality of the q_s variables. When we obtain fractional q_s variables in the linear solution, we branch by identifying the q_s variable with fractional value closest to a threshold value and round it down to the nearest integer in one branch and round it up to the nearest integer in the other branch. However, even if all q_s are integer, we do not necessarily have integer values on the cycles.

The third approach, BB-3, utilizes the fact that the number of cycles in the optimal solution is integer. Hence, we branch on the slack variable in constraint (2b). This branching strategy alone cannot guarantee finding integer solutions for the design-cycle variables.

We have implemented a depth-first tree-exploration strategy with backtracking. Within this approach, we have tested the three different branching strategies extensively to find the best

combination to use. The computational result section presents the results from the experimentation of the different branching strategies.

2.5 An acceleration technique for upper bound computations

The upper bound is updated each time a new best integer solution is found. In order to encourage fathoming, it is of high importance to obtain good integer solutions early in the traversal of the tree. We introduce an acceleration technique with the aim of finding integer solutions quickly through a depth-first search strategy and thus improve the performance of the B&P algorithm.

From the solution of the linear RMP, we fix the cycle variables g_k with fractional values above a given threshold to one, or if no variables g_k have fractional values above the threshold, the cycle variable with highest fractional value is fixed to one. Then, new columns are generated until the solution of the RMP cannot be improved further. This procedure is repeated until an integer solution has been found, the solution is worse than the current best integer solution, or the problem becomes infeasible. The success of this technique depends on the value of the threshold. If it is set too low, we expect to find a poor integer solution. However, a high value of the threshold may slow down the process of fixing columns to 1. It is possible to change this threshold value during the solution process.

This acceleration technique can be used at the end of some branch-and-bound nodes in the tree or at the end of all nodes. It is a trade-off between finding a good upper bound and the extra computational time needed by this technique. Due to our testing, we have chosen to perform the technique at the end of each branch-and-bound node. When the technique is completed at each node, we "unfix" the involved variables. In addition, we delete the columns generated during the fixing procedure, before branching further on in the tree. We may restrict the application of the acceleration technique by defining a maximum time for it at each node.

3 Computational Study

We present a computational study based on the branch-and-price (B&P) algorithm introduced in Section 2 for the SNDAM problem. The algorithms are programmed in C++ and run on 64 bits computers with 3 GHz processors and 8 GB RAM running under Rock Cluster v 4.2.1 operating system. The restricted master problems are solved with the LP-solver of XPRESS Optimizer v 17.1. We first discuss implementation and calibration issues in Section 3.1, before introducing the data sets used in Section 3.2. Finally, computational results appear in Section 3.3.

3.1 Implementation and Calibration

The computational study explores three different solution approaches, denoted "A", "B", and "C" mainly targeted at different instance sizes. The three solution approaches are developed through extensive testing in a calibration phase, and the parameter setting is based on what was observed to yield reasonable performance over a set of instances. We summarize significant implementation issues and parameter values in Table 1.

For all approaches, we implement a depth-first strategy in exploring the B&P tree, and set a maximum running time of 10 hours (36000 seconds), which is the same as in the computational study in Andersen *et al.* (2009b) and thus facilitates comparisons. In solving the RMP, we save the basis before adding columns, and we reoptimize with the primal simplex from the saved basis once new columns are added. As pointed out in Sections 2.2 and 2.3, it is computationally cheap to add more than one cycle or path when the subproblems are solved. There is a trade-off between the gains of saving iterations as a consequence of adding multiple columns, versus the increased

computational time from having more columns in the master problem. There are a significant amount of commodities in our data sets, and we therefore allow inclusion of at most one path for each commodity in an iteration of the column generation. However, because of the many commodities, it makes sense to include multiple cycles in each iteration if there are several cycles with negative reduced costs. We allow most cycles in solution approach C, which is targeted at the largest instances

Table 1. Implementation issues and parameter values.

Issues and parameters	Approach A	Approach B	Approach C
Search strategy	Depth first		
Maximum search time	36000 seconds		
How often is the master problem solved?	All subproblems are solved for given dual information before the master problem is resolved		
Strategy for solving RMP at each BB-node	Save basis before adding columns, reoptimize with primal simplex from this basis when columns have been added		
Branching strategies	BB-2/BB-1	(BB-3) / BB-2 / BB-1	
How often are strong forcing constraints (SFC) generated?	All nodes	Nodes where lower bounds are updated	
How long do we keep on generating strong forcing constraints at a node?	As long as they are violated	No new iterations after 18000 seconds or if improvement < 0.1%	
Should we keep SFC in the problem after generation?	Yes		No
Max number of paths generated in each iteration	One per commodity		
Max no of new cycles in each col.gen. iteration in BB-node	5		10
Max no of new cycles in each col gen iteration in acc. techn.	1		2
Min and max threshold for fixing cycle var to 1 in acc. technique	0.55-0.7		0.55-0.65
Simplex tolerance level and no of iterations without improvement	10E-6 / 50	10E-6 / 10	
Maximum time for acceleration technique at each BB-node	No restrictions	30 min	

For branching, the calibration phase indicated that one should branch on either service frequency (BB-2) or fleet size (BB-3) slack variables, before branching on arcs (services) in the underlying network. In solution approach A, we first branch on BB-2 until all service frequency slack variables take integer values, and then switch to BB-1. In solution approaches B and C, we use BB-3, then BB-2, and again BB-1 if all slack variables take integer values, but we also implement these approaches with BB-2 and BB-1 only.

An important difference between the three approaches is the role of the strong forcing constraints. These constraints improve the lower bounds and could also contribute to better branching decisions and better performance of the acceleration technique, since including such constraints removes parts of the solution space that is feasible for the weak linear relaxation of the problem, but not for the original mixed integer version. The generation of strong forcing constraints is time-consuming for large problem instances, however, due to the many constraints that could be added.

In solution approach A we test the effect of generating strong forcing constraints at all the nodes of the tree. The limit for fixing cycles for the acceleration technique described in Section 2.5 is initially set to 0.7, and sequentially reduced to 0.55. At most 5 cycles may be added at each iteration of the column generation procedure. To avoid instability in the column generation, a tolerance level of 10^{-6} for relative change in objective function value is introduced. If the objective function does not improve by this value in fifty iterations, we stop the iterations and the algorithm continues.

In solution approach B, strong forcing constraints are generated only at the nodes where the lower bounds are updated, which should allow the exploration of more nodes within the time limit. The tolerance level for simplex is reduced to 10 iterations without a relative objective function value improvement of 10^{-6}.

Solution approach C is targeted at huge problem instances with several hundred commodities. The acceleration technique starts with a limit for fixing cycles at 0.65. At most 10 cycles with negative reduced cost may be added at each iteration of the column generation algorithm, but only two within the acceleration technique. Strong forcing constraints are generated only when the lower bound is updated. Moreover, the strong forcing constraints are removed from the problem as soon as the bound is updated, to facilitate faster exploration of the search tree. For both solution approaches B and C, we include the opportunity to stop this process if the time exceeds 18000 seconds or if the improvement in the objective function value at the last iteration was below 0.1%.

For all solution approaches, we strengthen the calculation of lower bounds by rounding $\sum_{k \in K} g_k$ up and down to the nearest integer values. In particular, if the rounding down yields an infeasible problem, this contributes to a stronger bound and thus a smaller optimality gap. Furthermore, for all solution approaches, the tolerance level is increased to 10^{-5} and five iterations at nodes where we cannot improve the lower bound, and also within the acceleration technique.

3.2 Data Sets

Andersen *et al.* (2009b) addressed SNDAM problem instances with all design-cycles and commodity paths generated a priori with the CPLEX MIP-solver. We test the B&P algorithm on a few of these instances (1 to 5 in Table 2), but the computational study is focused on significantly larger instances than what can be solved with a priori generation. The new instances are based on a real-life case study in rail transportation planning, but are randomly generated with increased numbers of terminals, services, time periods, and commodities to challenge the branch-and-price algorithm. The instances are available from http://www.iot.ntnu.no/users/mc. We present in Table 2 the dimensions of the instances. Columns two to four give the number of terminals and services in the physical network, as well as the number of time periods, respectively. We present the corresponding number of possible service departures and holding arcs in column five and the number of commodities in column six. A *size indicator* for problem dimension is presented in the seventh column, computed as the product of the number of possible service departures, time periods, and commodities and divided by 1000. This measure gives a fair description of the relative dimensions of the instances. As pointed out in Section 3.1, initial testing has suggested that different approaches should be applied depending on the sizes of the instances. We thus indicate in the last column the solution approaches that are applied to each instance.

Instances 1-5 are small and we apply solution approaches A and B. Only approach B is applied to instances 6 to 12, for which the size indicator is in the range 120 to 450, and significantly larger than for instances 1-5. Finally, for the largest instances 13-15, the size indicator exceeds 1000 and we apply solution approaches B and C. For each of the reasonably large instances 6-12, we generate five different data sets to assess the variability of the algorithm performance.

3.3 Results

In this section we present the experimental results for the problem instances defined in Table 2. We present results for problems that have previously been solved with a priori generation of columns (Andersen et al., 2009b) in Section 3.3.1, while Section 3.3.2 presents results for new problem instances. More detailed results are presented in Annex A for all fifteen instances.

Table 2. Dimensions of problem instances.

Problem id #	Terminals $\|N\|$	Services $\|S\|$	Time periods T	Service + holding arcs	Commodities $\|C\|$	Size indicator	Solution approaches
1	5	10	20	200 + 100	20	4	A and B
2	5	15	20	300 + 100	25	8	A and B
3	5	15	25	375 + 125	25	9	A and B
4	5	15	15	225 + 75	100	23	A and B
5	5	15	15	225 + 75	200	45	A and B
6	5	15	40	600 + 200	200	120	B
7	5	15	50	750 + 250	400	300	B
8	7	30	30	900 + 210	200	180	B
9	7	30	30	900 + 210	400	360	B
10	7	30	50	1500 + 350	300	450	B
11	10	40	30	1200 + 300	200	240	B
12	10	50	30	1500 + 300	100	150	B
13	10	50	30	1500 + 300	1000	1500	B and C
14	7	30	60	1800 + 420	800	1440	B and C
15	10	50	50	2500 + 500	400	1000	B and C

3.3.1 Instances solved with a priori generation of columns

Table 3 displays the results for instances 1-5. Two rows are presented for each instance, one for each solution approach A and B (the latter with branching strategies BB-2/BB-1). In each case, we report in column three and four the lower bound and the best MIP solution at termination, which occurs after 10 hours of CPU time or once the algorithm has found a provably optimal integer solution. The fifth column displays either the solution time in CPU seconds for finding the provably optimal solution with B&P, or the optimality gap after 10 hours of CPU time. The last two columns display the best integer solution obtained with a priori enumeration of cycles and paths and the corresponding solution time or remaining optimality gap (Andersen *et al.*, 2009b). The a priori enumeration was implemented on different computers but with comparable specifications and a maximum running time of 10 hours. The numbers of constraints and columns generated and the shares of time used for the master problem, subproblems, and management are presented in Annex A.

Table 3. Results for instances previously addressed with a priori generation of columns.

		Branch-and-price algorithm			A priori enumeration	
Instance	Solution approach	Lower bound	MIP solution	Solution time (sec) / Optimality gap	MIP solution	Solution time (sec) / Optimality gap
1	A	48 838	48 838	237	48 838	9
	B	48 838	48 838	161		
2	A	52 156	52 156	28 578	52 156	1235
	B	52 156	52 156	16 508		
3	A	47 805	47 805	921	47 805	143
	B	47 805	47 805	1 271		
4	A	171 532	174 687	1.80 %	174 233*	0.7%
	B	171 532	174 687	1.80 %		
5	A	378 347	381 040	0.71 %	381 533*	0.5%
	B	378 347	380 999	0.70 %		

**For these instances, the a priori approach did not return a provably optimal solution within 10 hours.*

The results reported in Table 3 are very similar for approaches A and B. For instances 1-3, the B&P algorithm returned provably optimal integer solutions with both solution approaches. For instances 1 and 3, the solutions were obtained reasonably quickly, while instance 2 required around 16500 and 28600 seconds, respectively. As expected, a top commercial MIP solver performed significantly faster when all variables were generated a priori. It is nevertheless good news that the B&P algorithm returns provably optimal integer solutions for problems where these optimal solutions are known. No method returned provably optimal solutions within 10 hours of CPU time for instances 4 and 5. For instance 4, the integer solutions obtained were slightly better with a priori enumeration, and the gap was also smaller. For instance 5, both approaches A and B of the branch-and-price algorithm returned better integer solutions than what was obtained with a priori enumeration. However, the remaining optimality gap after 10 hours is smaller with a priori enumeration. The stronger focus on bounding in a commercial MIP-solver explains most of these observations. To conclude this phase of the experimentation, the proposed B&P algorithm solves small instances to proven optimality, and yields solutions of similar quality as a priori enumeration for instances at the limit of what could be handled with this approach (Andersen *et al.*, 2009b).

3.3.2 Larger instances

We present results for problem instances 6-12 in Table 4. All integer solutions were obtained using the acceleration technique presented in Section 2.5. No instance was solved to proven optimum within 10 hours of CPU time. There are five rows for each instance in the table, representing the five data sets for each instance. The results in Table 4 were obtained with branching strategies BB-2/BB-1, which appeared to produce the smallest optimality gaps. Results obtained with BB-3/BB-2/BB-1 are presented in Annex B.

The third column of Table 4 presents the lower bounds for the original problem at termination, which in all cases corresponds to the lower bound obtained at the root node, because the depth-first search did not return to the root node within 10 hours for any of these instances. Best integer solutions and optimality gaps are reported in the next two columns, followed by the average gap for each instance. The following columns report the node number where the best integer solution was found and the elapsed time, respectively. In the last two columns, we report the number of branch-and-price nodes visited during the tree search, and the time needed to solve the root node. The numbers of constraints and columns generated and the shares of time used for the master problem, subproblems, and management are presented in Annex A.

From Table 4 we observe that the optimality gaps range from 1.2% to 14.7%, and the averages for each instance are in the range 3.8% - 7.3%. The results are in other words reasonably stable with optimality gaps around 4-7%, but with a few outliers. The optimal integer solutions are not known for these problems. In testing smaller but similar problems (Andersen *et al.*, 2009b) we observed, however, that the gaps between the strong linear relaxation at the root node and the optimal integer solution were in the range 0-9%, with an average of about 3%. It is thus likely that the integer solutions reported in Table 4 represent near-optimal solutions to the instances. Similar figures appear in computational studies of similar problems, e.g., (Ghamlouche *et al.*, 2004) and (Pedersen *et al.*, 2009).

We report results obtained for instances 13-15 using branching strategy BB-2/BB-1 in Table 5. Results obtained with BB-3/BB-2/BB-1 are presented in Annex B. The columns of Table 5 are identical to those of Table 4, except that we also indicate whether solution approach B or C was used and we do not present averages for each instance. The number of constraints and columns generated and the shares of time used for the master problem, subproblems, and management are presented in Annex A.

Table 4. Results from model runs for instances 6-12 using branching strategies BB-2/BB-1. Maximum computational time is 10 hours.

Instance	Data set	Lower bound	Best MIP-solution	Optimality gap	Average gap	Best MIP-node	Best MIP-time	Nodes visited	Root time
6	A	199 276	207 073	3.9 %	3.8 %	3	1 837	582	1 589
	B	76 979	79 850	3.7 %		10	14 363	127	5 210
	C	211 662	222 962	5.3 %		48	10 394	564	1 383
	D	242 283	254 038	4.9 %		42	6 317	1 224	410
	E	334 026	337 970	1.2 %		5	185	1 657	83
7	A	380 049	399 942	5.2 %	4.5 %	3	2 465	93	1 870
	B	472 281	492 329	4.2 %		3	1 823	135	1 252
	C	132 022	137 247	4.0 %		9	26 687	14	6 595
	D	178 064	188 468	5.8 %		3	8 811	32	6 973
	E	465 997	481 169	3.3 %		18	24 797	29	6 717
8	A	416 312	438 209	5.3 %	4.4 %	4	3 113	955	449
	B	156 979	163 073	3.9 %		3	10 134	269	4 172
	C	416 949	436 242	4.6 %		31	12 990	1 140	350
	D	360 192	376 045	4.4 %		28	13 120	915	2 247
	E	360 466	373 555	3.6 %		2	2 412	1 338	2 181
9	A	234 843	248 498	5.8 %	4.3 %	3	14 418	10	7 814
	B	346 985	363 165	4.7 %		3	5 269	45	3 310
	C	724 181	749 700	3.5 %		15	13 354	128	2 403
	D	743 061	766 424	3.1 %		2	735	358	512
	E	363 332	378 938	4.3 %		8	17 188	28	3 854
10	A	460 244	480 772	4.5 %	5.6 %	1	14 551	5	16 506
	B	276 288	293 161	6.1 %		1	26 134	2	28 404
	C	286 209	306 029	6.9 %		1	31 314	2	31 713
	D	498 922	526 916	5.6 %		3	16 444	9	10 905
	E	504 950	529 821	4.9 %		8	27 518	12	9 216
11	A	301 161	314 763	4.5 %	7.3 %	6	21 917	13	5 404
	B	585 457	611 721	4.5 %		7	5 508	91	936
	C	183 394	210 386	14.7 %		1	12 394	3	14 634
	D	189 225	201 251	6.4 %		15	34 971	15	4 238
	E	412 463	439 784	6.6 %		3	6 728	21	3 835
12	A	157 695	161 507	2.4 %	4.0 %	3	9 834	22	5 788
	B	215 855	217 811	0.9 %		3	4 824	368	2 489
	C	142 153	152 875	7.5 %		3	13 768	6	7 466
	D	261 486	277 024	5.9 %		18	18 161	37	2 939
	E	284 210	293 856	3.4 %		9	5 453	292	388

Table 5 displays that integer solutions were found for all instances with solution approach C, the gaps ranging from 13. 5% to 21.5%. With solution approach B, integer solutions were found for instances 13 and 14, but not for instance 15. When integer solutions were found, the resulting optimality gaps were 4,9% and 11.2%, which is compatible with the results reported in Table 4. Thus, as long as solution approach B worked, good solutions were obtained, while with solution approach C the remaining optimality gaps were considerably larger. The main difference between

approaches B and C is that the generated strong forcing constraints are kept in the problem with approach B, which again "drive" the acceleration technique in a desirable direction. But the cost of this benefit is that the time associated with solving the problem increases, and there is thus a risk that no solution is obtained for the largest instances

Table 5. Results from model runs for instances 13-15. Maximum computational time is 10 hours.

Instance	Solution approach	Lower bound	Best MIP-solution	Optimality gap	Best MIP-node	Best MIP-time	Nodes visited	Root time
13	B	733 766	769 808	4.9 %	1	22 631	2	23 833
	C	733 766	851 055	16.0 %	3	20 508	6	12 699
14	B	645 452	717 475	11.2 %	1	27 214	2	29 752
	C	644 264	782 793	21.5 %	1	32 980	2	35 016
15	B	632 116	-	-	-	-	2	35 937
	C	624 306	708 632	13.5 %	2	29 687	3	24 353

3.3.3 Comparison with Simpler Approaches

The aim of this paper is to develop an exact solution method for the SNDAM problem that could handle larger instances than what was solved by a priori column generation (Andersen *et al.*, 2009b). In this section, we compare the B&P algorithm with two simpler approaches, where column generation is performed once (corresponding to the root node), and then a commercial MIP solver is used to solve the problem with the resulting set of columns.

Approach "MIP-solver I" solves the root node in the same way as the B&P method, but without the acceleration technique. It then exports the problem to Xpress-Optimizer, which is allowed to use the remaining time from the initial 36000 seconds. Approach "MIP-solver II" includes the acceleration technique from the root node to allow more columns into the problem, and keeps those columns in the problem exported to Xpress-Optimizer in the same way as in approach "MIP-solver I". However, if the acceleration technique has identified integer solutions, these are used for cut-off to enhance the performance of the commercial solver. A comparison of the optimality gaps obtained with B&P and the two simpler approaches is presented in Figure 5. Only instances 1-12 are illustrated to maintain the readability of the figure. The averages over the five data sets for each instance are presented for instances 6-12. Detailed integer solutions for all instances are presented in Annex C.

The results illustrated in Figure 5 indicate that the B&P algorithm generally performs significantly better than the two simpler approaches. "MIP-solver II" is the second best, mainly because it uses integer solutions from the acceleration technique applied at the root node. The "MIP-solver I" approach yields the largest gaps and suffers from huge fluctuations from instance to instance. We thus conclude that a full B&P algorithm offers a more reliable solution approach than the considered simpler approaches.

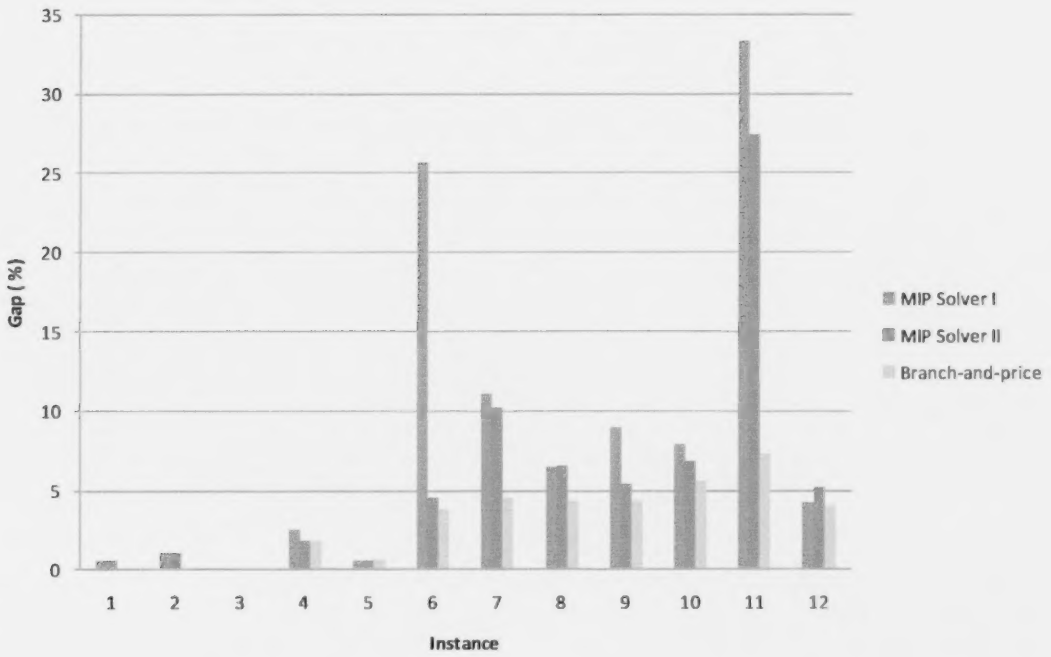

Figure 5. Optimality gaps obtained with the B&P algorithm and the two MIP-solver approaches.

4 Concluding Remarks

We presented a branch-and-price (B&P) algorithm for addressing the service network design problem with asset management constraints (SNDAM). Such problems arise when decisions on vehicle management are considered jointly with service network design issues, and represent an important potential for improved planning of transportation systems.

The proposed method integrates two column-generation subproblems for integer cycle-design and continuous flow-path variables, a combination of branching strategies, a mechanism to dynamically add violated strong linear relaxation cuts, and an acceleration technique to assist in rapidly identifying integer solutions. Experimental studies have shown that the proposed algorithm performs very satisfactorily. It is comparable for small problem instances with a commercial solver applied with a priori complete enumeration of cycle and path variables and is able to address much larger instances. The performance of the B&P algorithm was compared to two simpler approaches based on column generation in the root node of the tree combined with the use of a commercial MIP-solver. The experiments showed that the B&P algorithm performed significantly better than the simpler appraoches, and the experiments thus indicate that a tree exploration is required to identify high-quality solutions.

Future work will focus on enhancing the proposed method. Interesting perspectives are offered by the improvement of the acceleration technique, for instance by reversing earlier variable-fixing decisions to foster diversity, the development of tighter lower bounds, the introduction of stronger cuts, e.g. (Chouman *et al.*, 2009), and the design of parallel computing strategies.

Acknowledgements

This work has received financial support from The Norwegian Research Council through the Polcorridor Logchain project. Partial funding has also been supplied by the Natural Sciences and Engineering Research Council of Canada (NSERC) through its Discovery and Industrial Research Chair programs. While working on this project, Dr. Teodor Gabriel Crainic was NSERC Industrial Research Chair in Logistics Management, ESG UQAM (Canada) and Adjunct Professor at Molde University College and the department of Computer Science and Operations Research of the Université de Montréal.

References

Agarwal, R., Ö. Ergun. 2008. Ship scheduling and network design for cargo routing in liner shipping. *Transportation Science* 42, 175-196.

Andersen, J., T.G. Crainic, M. Christiansen. 2009a. Service network design with management and coordination of multiple fleets. *European Journal of Operational Research* 193(2), 377-389.

Andersen, J., T.G. Crainic, M. Christiansen. 2009b. Service network design with asset management: Formulations and comparative analyzes. *Transportation Research Part* C, 17, 197-207.

Balakrishnan, A., T.L. Magnanti, P. Mirchandani. 1997. Network Design. In *Annotated bibliographies in combinatorial optimization*, Dell'Amico, M., F. Maffoli F, S. Martello (eds), John Wiley & Sons: New York, NY.

Barnhart, C., R.R. Schneur. 1996. Air network design for express shipment service. *Operations Research* 44, 852- 863.

Barnhart, C., E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, P.H. Vance. 1998. Branch-and-price: Column generation for solving huge integer programs. *Operations Research* 46, 316–329.

Christiansen, M., K. Fagerholt, B. Nygreen, D. Ronen. 2007. Maritime transportation. In: Barnhart, C., Laporte, G. (Eds.), *Transportation*, volume 14 of *Handbooks in Operations Research and Management Science*, North-Holland, Amsterdam, pp. 189-284.

Chouman, M., T.G. Crainic, Gendron, B. 2009, A Cutting-Plane Algorithm for Multicommodity Capacitated Fixed-Charge Network Design, Publication CIRRELT-2009-03, Interuniversity Research Centre on Enterprise Networks, Transportation and Logistics, Université de Montréal, Canada.

Cordeau, J.-F., P. Toth, D. Vigo. 1998. A survey of optimization models for train routing and scheduling. *Transportation Science* 32, 380-404.

Crainic, T.G. 2000. Service network design in freight transportation. *European Journal of Operational Research* 122, 272-288.

Crainic, T.G. 2003. Long-haul freight transportation. In: Hall, R.W. (Ed.), *Handbook of Transportation Science*, 2nd Edition, Kluwer Academic Publishers, New York, pp. 451-516.

Crainic, T.G., K.H. Kim. 2007. Intermodal transportation. In: Barnhart, C., Laporte, G. (Eds.), *Transportation*, volume 14 of *Handbooks in Operations Research and Management Science*, North-Holland, Amsterdam, pp. 467-537.

Desaulniers, G., J. Desrosiers, M.M. Solomon (Eds.). 2005. *Column generation*. GERAD 25th Anniversary Series. Springer.New York, NY.

Ghamlouche, I., T.G. Crainic, M. Gendreau. 2004. Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design. *Annals of Operations Research* 131, 109-133.

Kim, D., C. Barnhart, K. Ware, G. Reinhardt. 1999. Multimodal express package delivery: a service network design application. *Transportation Science* 33, 391-407.

Lai, M.F., H.K. Lo. 2004. Ferry service network design: optimal fleet size, routing and scheduling. *Transportation Research A* 38, 305-328.

Lübbecke, M.E., J. Desrosiers. 2005. Selected topics in column generation. *Operations Research* 53, 1007-1023.

Magnanti, T.L., R.T. Wong. 1984. Network design and transportation planning: models and algorithms. *Transportation Science* 18, 1-55.

Pedersen, M.B., T.G. Crainic. 2007. Optimization of intermodal freight train service schedules on train canals. Publication CIRRELT-2007-51. Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal.

Pedersen, M.B., T.G. Crainic, O.B.G. Madsen. 2009. Models and tabu search metaheuristics for service network design with asset-balance requirements. Transportation Science, 43, 158-177.

Sigurd, M., N.L. Ulstein, B. Nygreen, D.M. Ryan. 2005. Ship scheduling with recurring visits and visit separation requirements. In: *Column generation*, Desaulniers, G., J. Desrosiers, M.M. Solomon (Eds.), GERAD 25th Anniversary Series. Springer. New York, NY, pp. 225-245.

Smilowitz, K.R., A. Atamtürk, C.F. Daganzo. 2003. Deferred item and vehicle routing within integrated networks. *Transportation Research E* 39, 305-323.

Annex A. Detailed results

The following table displays measures related to the performance of the branch-and-price algorithm with branching strategy BB-2/BB-1. Columns three and four indicate the numbers of strong forcing constraints generated at the root node and during the entire tree exploration, respectively. Columns five to ten display the number of cycles and paths generated in the model runs. For the root node, cycles and paths generated within the acceleraing technique ("Root acc.") are presented separate from the cycles and paths generated in the standard node solving ("Root std"). The last three columns display the percentage of the total CPU seconds required to address 1) the restricted master problem, 2) the cycle and path generation subproblems, and 3) everything else in the code which is not direct Xpress Optimizer solution time for the master problem or time needed to solve the subproblems.

		Generated strong forcing constraints		Generated cycles			Generated paths			Distribution of time use		
Inst.	Solution approach	Root	Tree search	Root std.	Root acc.	Tree search	Root std.	Root acc.	Tree search	Master	Sub	Admin*
1	A	196	215	218	14	9 022	372	79	17 706	95 %	2 %	3 %
	B	196	7	218	14	7 390	372	79	13 147	95 %	2 %	3 %
2	A	315	795	463	214	398 371	723	240	379 456	97 %	1 %	2 %
	B	315	94	463	214	347 197	723	240	239 262	98 %	1 %	2 %
3	A	462	75	779	1 117	7 371	1 198	211	1 244	97 %	1 %	2 %
	B	484	13	784	803	8 735	1 205	125	1 398	97 %	1 %	2 %
4	A	313	1 109	296	33	216 421	987	178	840 866	94 %	1 %	5 %
	B	313	0	296	33	376 384	987	178	1 036 790	93 %	2 %	5 %
5	A	131	39 609	265	0	95 987	2 928	75	571 318	88 %	1 %	11 %
	B	131	0	265	0	289 203	2 928	75	1 312 073	83 %	4 %	13 %
6	A	768	0	4 035	1 199	24 529	2 263	444	11 917	99 %	0 %	1 %
	B	3 612	0	6 194	340	4 974	4 619	706	6 001	100 %	0 %	0 %
	C	729	0	5 040	672	21 564	2 208	341	12 957	99 %	0 %	1 %
	D	618	0	3 175	89	43 382	3 465	301	15 948	98 %	0 %	1 %
	E	537	0	1 050	251	39 383	4 261	180	19 656	97 %	0 %	2 %
7	A	1 035	0	4 740	491	18 617	6 801	435	21 294	96 %	0 %	4 %
	B	839	0	4 275	383	18 685	6 303	438	24 176	96 %	0 %	3 %
	C	4 299	0	6 628	38	1 622	5 037	699	9 460	99 %	0 %	1 %
	D	2 236	0	7 410	135	3 264	3 988	657	12 957	99 %	0 %	1 %
	E	3 093	0	6 370	172	4 992	7 762	605	12 949	99 %	0 %	1 %
8	A	1 082	0	2 087	244	28 519	4 095	805	47 587	97 %	0 %	2 %
	B	4 834	0	3 407	385	6 215	7 188	1 736	22 844	99 %	0 %	1 %
	C	1 141	0	1 829	92	34 714	4 360	823	45 531	96 %	1 %	3 %
	D	1 328	0	2 433	2 079	28 552	4 311	1 148	42 718	97 %	0 %	2 %
	E	1 250	0	2 474	2 053	33 507	4 199	1 352	34 661	99 %	1 %	0 %
9	A	7 576	0	3 045	433	1 058	11 153	3 493	24 538	99 %	0 %	1 %
	B	3 473	0	2 429	903	4 631	7 843	1 998	52 188	97 %	0 %	2 %
	C	1 160	0	2 171	1 974	21 551	9 821	1 072	55 507	93 %	1 %	6 %
	D	1 270	0	1 874	121	20 545	8 683	1 363	88 362	95 %	1 %	4 %
	E	3 537	0	2 616	1 072	5 158	8 273	2 580	45 307	98 %	0 %	2 %
10	A	1 834	0	9 455	227	691	5 814	655	2 840	99 %	0 %	1 %
	B	3 639	0	11 010	167	98	6 058	1 349	946	99 %	0 %	1 %
	C	3 663	0	11 385	103	56	6 538	1 185	588	99 %	0 %	1 %
	D	1 520	0	9 365	179	1 733	6 123	617	4 539	99 %	0 %	1 %
	E	1 478	0	8 115	430	2 365	6 083	735	6 229	99 %	0 %	1 %
11	A	4 198	0	3 860	399	1 813	8 227	3 672	33 371	99 %	0 %	1 %
	B	1 638	0	2 379	171	12 528	7 423	2 442	97 965	98 %	1 %	2 %

		Generated strong forcing constraints		Generated cycles			Generated paths			Distribution of time use		
Inst.	Solution approach	Root	Tree search	Root std.	Root acc.	Tree search	Root std.	Root acc.	Tree search	Master	Sub	Admin*
C	B	7 142	0	4 239	438	208	11 740	5 332	7 880	99 %	0 %	0 %
D	B	5 894	0	3 571	70	1 570	8 777	3 806	44 507	99 %	0 %	1 %
E	B	1 837	0	3 133	978	5 202	7 265	2 740	46 385	98 %	0 %	2 %
A	B	2 793	0	5 640	343	1 189	5 209	1 280	12 607	100 %	0 %	0 %
B	B	1 925	0	3 826	1 001	9 854	3 522	1 345	34 621	99 %	0 %	1 %
12 C	B	5 224	0	5 031	41	366	9 425	1 922	9 682	100 %	0 %	0 %
D	B	1 327	0	3 820	1 143	5 440	4 083	1 631	35 657	99 %	0 %	1 %
E	B	1 015	0	2 278	96	15 622	2 604	1 083	42 244	99 %	0 %	1 %
13	B	6 934	0	4 139	179	182	24 824	4 547	4 501	98 %	0 %	1 %
	C	6 947	0	4 896	2 059	9 908	21 235	821	4 123	93 %	1 %	6 %
14	B	3 212	0	6 830	463	280	15 850	1 807	1 835	98 %	0 %	2 %
	C	3 110	0	10 460	2 395	92	12 208	426	38	98 %	0 %	2 %
15	B	2 153	0	9 675	203	15	10 671	1 248	4	99 %	0 %	1 %
	C	1 310	0	13 040	882	1 870	8 797	797	1 216	98 %	0 %	2 %

Annex B. Results obtained by branching on fleet size (BB-3)

In the third column lower bound from the root node is presented. In columns four and five the best integer solutions and corresponding optimality gaps using branching strategies BB-2/BB-1 are repeated from the main text. Then in columns six to nine, we present best integer solutions and corresponding optimality gaps when also BB-3 is included. In columns six and seven, we report results with BB-3 rounding the fleet size obtained in the root node down to the nearest integer, while columns eight and nine contain results obtained while rounding the fleet size from the solution in the root node up to the nearest integer.

			Service freq. (BB-2/BB-1)		Incl BB-3 fleet size down		Incl BB-3 fleet size up	
Instance		Lower bound	Best MIP	Opt. gap	Best MIP	Opt. gap	Best MIP	Opt. gap
6	A	199 276	207 073	3.9 %	210 083	5.4 %	210 081	5.4 %
	B	76 979	79 850	3.7 %	81 024	5.3 %	81 024	5.3 %
	C	211 662	222 962	5.3 %	223 816	5.7 %	221 584	4.7 %
	D	242 283	254 038	4.9 %	254 648	5.1 %	254 475	5.0 %
	E	334 026	337 970	1.2 %	342 752	2.6 %	340 505	1.9 %
7	A	380 049	399 942	5.2 %	404 638	6.5 %	400 776	5.5 %
	B	472 281	492 329	4.2 %	493 830	4.6 %	493 830	4.6 %
	C	132 022	137 247	4.0 %	137 722	4.3 %	137 983	4.5 %
	D	178 064	188 468	5.8 %	188 052	5.6 %	189 056	6.2 %
	E	465 997	481 169	3.3 %	481 194	3.3 %	484 684	4.0 %
8	A	416 312	438 209	5.3 %	433 572	4.1 %	437 562	5.1 %
	B	156 979	163 073	3.9 %	181 292	15.5 %	163 655	4.3 %
	C	416 949	436 242	4.6 %	432 258	3.7 %	436 197	4.6 %
	D	360 192	376 045	4.4 %	374 852	4.1 %	373 687	3.7 %
	E	360 466	373 555	3.6 %	374 331	3.8 %	373 555	3.6 %
9	A	234 843	248 498	5.8 %	248 969	6.0 %	248 969	6.0 %
	B	346 985	363 165	4.7 %	367 062	5.8 %	363 946	4.9 %
	C	724 181	749 700	3.5 %	750 649	3.7 %	754 654	4.2 %
	D	743 061	766 424	3.1 %	766 704	3.2 %	763 246	2.7 %
	E	363 332	378 938	4.3 %	379 852	4.5 %	382 082	5.2 %
10	A	460 244	480 772	4.5 %	480 772	4.5 %	480 772	4.5 %
	B	276 288	293 161	6.1 %	292 854	6.0 %	292 854	6.0 %
	C	286 209	306 029	6.9 %	306 029	6.9 %	306 029	6.9 %
	D	498 922	526 916	5.6 %	525 194	5.3 %	531 406	6.5 %
	E	504 950	529 821	4.9 %	539 546	6.9 %	537 781	6.5 %
11	A	301 161	314 763	4.5 %	320 581	6.4 %	317 540	5.4 %
	B	585 457	611 721	4.5 %	611 721	4.5 %	611 721	4.5 %
	C	183 394	210 386	14.7 %	210 386	14.7 %	210 386	14.7 %
	D	189 225	201 251	6.4 %	202 874	7.2 %	202 874	7.2 %
	E	412 463	439 784	6.6 %	435 961	5.7 %	435 414	5.6 %
12	A	157 695	161 507	2.4 %	163 418	3.6 %	162 439	3.0 %
	B	215 855	217 811	0.9 %	221 747	2.7 %	218 104	1.0 %
	C	142 153	152 875	7.5 %	160 554	12.9 %	160 083	12.6 %
	D	261 486	277 024	5.9 %	268 739	2.8 %	274 160	4.8 %

	E	284 210	293 856	3.4 %	288 793	1.6 %	291 623	2.6 %
13	Sol B	733 766	769 808	4.9 %	769 808	4.9 %	769 808	4.9 %
	Sol C	733 766	851 055	16.0 %	817 693	11.4 %	850 540	15.9 %
14	Sol B	645 452	717 475	11.2 %	717 475	11.2 %	717 475	11.2 %
	Sol C	644 264	782 793	21.5 %	782 793	21.5 %	782 793	21.5 %
15	Sol B	632 116	-	-	-	-	-	-
	Sol C	624 306	708 632	13.5 %	717725	15.0 %	708427	13.5 %

Annex C. Integer solutions obtained with various approaches

The following table displays the values of the integer solutions obtained with the methods presented in this paper. The second and third columns display the solutions obtained using the "MIP-solver I" approach and the acceleration technique in the root node, respectively. The latter procedure is used for both the branch-and-price and the" MIP-solver II" approach. The fourth and the fifth columns thus indicate if branch-and-price and "MIP-solver II", respectively, improved the solution obtained in the root node. The last column sums up the experimental results indicating the solution method that yielded the best solution.

Instances		MIP-solver I	Root node	Branch-and-price	MIP-solver II	Best approach
1		49 127	-	48 838	49 127	Branch-and-price
2		52 685	53 209	52 156	52 685	Branch-and-price
3		47 805	47 805	-	-	All equal
4		175 819	174 687	-	-	Branch-and-price / MIP Solver II
5		380 565	383 238	380 999	380 363	MIP Solver II
6	A	-	210 081	207 073	-	Branch-and-price
	B	85 329	81 024	79 850	-	Branch-and-price
	C	229 948	223 816	222 962	-	Branch-and-price
	D	261 920	255 329	254 038	-	Branch-and-price
	E	337 282	342 752	337 970	337 131	MIP Solver II
7	A	420 619	404 638	399 942	-	Branch-and-price
	B	510 406	493 830	492 329	-	Branch-and-price
	C	146 863	143 154	137 247	-	Branch-and-price
	D	206 885	189 564	188 468	-	Branch-and-price
	E	509 786	-	481 169	583 943	Branch-and-price
8	A	441 750	448 387	438 209	442 073	Branch-and-price
	B	170 057	181 292	163 073	175 487	Branch-and-price
	C	442 660	-	436 242	442 054	Branch-and-price
	D	381 943	378 723	376 045	-	Branch-and-price
	E	380 568	374 331	373 555	-	Branch-and-price
9	A	279 878	248 969	248 498	-	Branch-and-price
	B	373 372	367 062	363 165	-	Branch-and-price
	C	751 967	758 346	749 700	750 088	Branch-and-price
	D	778 008	766 704	766 424	-	Branch-and-price
	E	397 062	393 020	378 938	-	Branch-and-price
10	A	494 476	480 772	-	-	Branch-and-price / MIP Solver II
	B	302 398	293 161	-	-	Branch-and-price / MIP Solver II
	C	312 334	306 029	-	-	Branch-and-price / MIP Solver II
	D	535 521	-	526 916	546 876	Branch-and-price
	E	535 240	539 823	529 821	-	Branch-and-price
11	A	337 979	320 581	314 763	-	Branch-and-price
	B	-	-	611 721	-	Branch-and-price
	C	220 875	210 386	-	-	Branch-and-price / MIP Solver II
	D	235 117	206 718	201 251	-	Branch-and-price

Instances		MIP-solver I	Root node	Branch-and-price	MIP-solver II	Best approach
	E	451 617	439 784	-	-	Branch-and-price / MIP Solver II
	A	161 137	167 208	161 507	-	MIP Solver I
	B	218 592	221 747	217 811	219 004	Branch-and-price
12	C	153 043	160 529	152 875	153 454	Branch-and-price
	D	274 960	322 829	277 024	277 571	MIP Solver I
	E	297 886	309 899	293 856	297 216	Branch-and-price
13	A	819 488	769 808	-	-	Branch-and-price / MIP Solver II
14	B	-	717 475	-	-	Branch-and-price / MIP Solver II
15	C	714 676	-	-	-	MIP Solver I